UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/698,820	10/31/2003	Matthew Englehart	MWS-062RCE2	1288

74321 7590 07/06/2009

LAHIVE & COCKFIELD, LLP/THE MATHWORKS
FLOOR 30, SUITE 3000
One Post Office Square
Boston, MA 02109-2127

EXAMINER	
CHEN, QING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
07/06/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/698,820	ENGLEHART ET AL.	
	Examiner	Art Unit	
	Qing Chen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *21 April 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the amendment filed on April 21, 2009.

2. **Claims 1-26** are pending.

3. **Claims 1-4, 11, 12, and 16** have been amended.

Response to Amendment

Claim Objections

4. **Claims 3, 13, and 18** are objected to because of the following informalities:

- **Claims 3, 13, and 18** contain a typographical error: The emphatic dash (–) in "user–selected" should be changed to a hyphen (-).

Appropriate correction is required.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. **Claims 1-8, 11-23, and 26** are rejected under 35 U.S.C. 103(a) as being unpatentable over "**Real-Time Workshop® User's Guide," January 1999 (hereinafter "RTW_UG")** in view of **US 2003/0056195 (hereinafter "Hunt")**.

As per **Claim 1**, RTW_UG discloses:

- providing an automatic code generator to create source code that implements functionality of said at least one graphical model and that corresponds to data referenced by said at least one graphical model *(see Figure 3-1; Chapter 3-3, "The Real-Time Workshop automates the task of building a stand-alone program from your Simulink model.")*; and

- providing a predefined storage class in said graphical modeling and execution environment, said predefined storage class specifying a first manner in which said automatic code generator creates said source code corresponding to said data referenced by said at least one graphical model in said graphical modeling and execution environment *(see Chapter 3-6, "All pages of the Simulation Parameters dialog box affect Real-Time Workshop code generation."; Chapters 3-15 to 3-16, "There is an Options button on the Real-Time Workshop page of the Simulation Parameters dialog box. Pressing this button opens a Code Generation Options dialog box that varies depending on which system target file configuration you selected in the System Target File Browser."; Chapter 3-19, "Storage Class — you can change the storage class of the selected variable.")*.

However, RTW_UG does not disclose:

- providing a user interface with a plurality of selectable parameters;

- defining a custom storage class in said graphical modeling and execution environment utilizing parameters selected by a user from said plurality of selectable parameters, said custom storage class specifying a second manner in which said automatic code generator creates source code corresponding to said data referenced by said at least one graphical model in

said graphical modeling and execution environment, said second manner differing from said first manner; and

- generating source code implementing said functionality of said at least one graphical model using said automatic code generator, said generating comprising:

- using said custom storage class to generate source code corresponding to said data referenced by said at least one graphical model.

Hunt discloses:

- providing a user interface with a plurality of selectable parameters *(see Figure 2; Paragraph [0049], "The code generator tool of the preferred embodiment of the present invention is embodied as an application program that presents the user with a Graphical User Interface (GUI) that can be used to easily input meta data about an object model.")*;

- defining a custom storage class in a graphical modeling and execution environment utilizing parameters selected by a user from said plurality of selectable parameters, said custom storage class specifying a second manner in which an automatic code generator creates source code corresponding to data referenced by at least one graphical model in said graphical modeling and execution environment, said second manner differing from a first manner *(see Paragraph [0006], "An object is defined by a user entering meta data defining the object and the object's relationships with objects stored in the common object repository. Source code is created from the meta data, wherein the source code defines the object and the object's relationships."; Paragraph [0050], "Turning now to FIG. 1, by way of an overview, the user of the code generator 100 ... of the present invention enters meta-data 102 into the code generator 100's GUI."; Paragraph [0083], "Consider now the entry of the meta data used in connection with the*

present invention. Such information representing the meta data is input via the GUI such as illustrated in FIGS. 2-6."; Paragraph [0088], "When the user is ready to build the class, the "Generate" button 218 can be selected, for example by clicking with a mouse."); and

- generating source code implementing functionality of said at least one graphical model using said automatic code generator *(see Paragraph [0092], "Referring now to FIG. 8, process 370 illustrates how the user interacts with the code builder tool 100 of the present invention starting at 380 where the code builder application is started. At 384 the builder dialogue 150 is instantiated to bring dialog 150 to the users display.")*, said generating comprising:

- using said custom storage class to generate source code corresponding to said data referenced by said at least one graphical model *(see Paragraph [0092], "The user selects an object [custom storage class] from the meta data at 388 and control passes to 392 where the user determines if the source code is to be created for the object. If the user chooses to create source code at this point (392), the source code is created at 396.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to modify Real-Time Workshop® to include providing a user interface with a plurality of selectable parameters; defining a custom storage class in said graphical modeling and execution environment utilizing parameters selected by a user from said plurality of selectable parameters, said custom storage class specifying a second manner in which said automatic code generator creates source code corresponding to said data referenced by said at least one graphical model in said graphical modeling and execution environment, said second manner differing from said first

manner; and generating source code implementing said functionality of said at least one graphical model using said automatic code generator, said generating comprising: using said custom storage class to generate source code corresponding to said data referenced by said at least one graphical model. The modification would be obvious because one of ordinary skill in the art would be motivated to allow the user to define a custom storage class that applies the same user edits to model data without introducing errors during automatic generation of source code *(see Hunt – Paragraph [0004])*.

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and RTW_UG further discloses:

- providing a view of salient aspects of said source code generated by said automatic code generator utilizing said user-selected parameters *(see Chapter 11-6, "MdlStart Function Custom Code dialog box")*.

As per **Claim 3**, the rejection of **Claim 2** is incorporated; however, RTW_UG does not disclose:

- changing said user-selected parameters for said custom storage class in said user interface; and

- adjusting said source code generated by said automatic code generator to reflect said change in said user-selected parameters.

Hunt discloses:

- changing user-selected parameters for a custom storage class in a user interface *(see Paragraph [0050], "When a change to the object model needs to be made, the user can simply update the meta data through the GUI and regenerate all the files associated with the object model.")*; and

- adjusting source code generated by an automatic code generator to reflect said change in said user-selected parameters *(see Paragraph [0050], "When a change to the object model needs to be made, the user can simply update the meta data through the GUI and regenerate all the files associated with the object model.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to modify Real-Time Workshop® to include changing said user-selected parameters for said custom storage class in said user interface; and adjusting said source code generated by said automatic code generator to reflect said change in said user-selected parameters. The modification would be obvious because one of ordinary skill in the art would be motivated to ascertain that the generated source code is concurrent with the most recent changes made by the user to the custom storage class parameters.

As per **Claim 4**, the rejection of **Claim 3** is incorporated; and RTW_UG further discloses:

- displaying salient aspects of said adjusted source code in said view of salient aspects of said source code *(see Chapter 11-6, "MdlStart Function Custom Code dialog box")*.

As per **Claim 5**, the rejection of **Claim 2** is incorporated; and RTW_UG further discloses:

- wherein said view of salient aspects of said source code automatically generated includes at least one token, said at least one token being symbolically representative of a non-displayed segment of said source code *(see Chapter 3-23, "Template makefiles are essentially makefiles in which certain tokens are expanded to create a makefile for your model (model.mk). model.mk is created from the template makefile specified in the Real-Time Workshop page of the Simulation Parameters dialog box by copying each line from the template makefile and expanding the tokens of Table 3-4.").*

As per **Claim 6**, the rejection of **Claim 1** is incorporated; however, RTW_UG does not disclose:

- wherein said custom storage class declares macros for instances of constant data.

Hunt discloses:

- wherein a custom storage class declares macros for instances of constant data *(see Paragraph [0050], "This meta-data represents a hierarchy of interfaces that include attributes and operations and is stored in a SQL compliant database in the preferred embodiment. From this data, the current embodiment of the code generator generates the following seven C++ files plus four database SQL files where applicable: ...").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to modify Real-Time Workshop® to include wherein said custom storage class declares macros for

instances of constant data. The modification would be obvious because one of ordinary skill in the art would be motivated to allow the user to specify commonly-used programming features when defining a custom storage class.

As per **Claim 7**, the rejection of **Claim 1** is incorporated; however, RTW_UG does not disclose:

- wherein said custom storage class declares variables for instances of constant data.

Hunt discloses:

- wherein a custom storage class declares variables for instances of constant data *(see Paragraph [0050], "This meta-data represents a hierarchy of interfaces that include attributes and operations and is stored in a SQL compliant database in the preferred embodiment. From this data, the current embodiment of the code generator generates the following seven C++ files plus four database SQL files where applicable: ...").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to modify Real-Time Workshop® to include wherein said custom storage class declares variables for instances of constant data. The modification would be obvious because one of ordinary skill in the art would be motivated to allow the user to specify commonly-used programming features when defining a custom storage class.

As per **Claim 8**, the rejection of **Claim 1** is incorporated; however, RTW_UG does not disclose:

- wherein said user-selected parameters control at least one of a manner in which automatically generated source code is defined, declared, accessed and addressed.

Hunt discloses:

- wherein user-selected parameters control at least one of a manner in which automatically generated source code is defined, declared, accessed and addressed *(see Figure 2; Paragraph [0049], "The code generator tool of the preferred embodiment of the present invention is embodied as an application program that presents the user with a Graphical User Interface (GUI) that can be used to easily input meta data about an object model."; Paragraph [0050], "Turning now to FIG. 1, by way of an overview, the user of the code generator 100 ... of the present invention enters meta-data 102 into the code generator 100's GUI."; Paragraph [0083], "Consider now the entry of the meta data used in connection with the present invention. Such information representing the meta data is input via the GUI such as illustrated in FIGS. 2-6.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to modify Real-Time Workshop® to include wherein said user-selected parameters control at least one of a manner in which automatically generated source code is defined, declared, accessed and addressed. The modification would be obvious because one of ordinary skill in the art would be motivated to allow the user control over how source code is generated from the custom storage class.

As per **Claim 11**, the rejection of **Claim 1** is incorporated; however, RTW_UG does not disclose:

- creating a separate header file with said automatic code generator in response to said selection of one of said plurality of user-selected parameters.

Hunt discloses:

- creating a separate header file with an automatic code generator in response to a selection of one of a plurality of user-selected parameters *(see Paragraph [0051], "1. A hierarchy of C++ Interface definition files shown as interface.h header files 106 ...")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to modify Real-Time Workshop® to include creating a separate header file with said automatic code generator in response to said selection of one of said plurality of user-selected parameters. The modification would be obvious because one of ordinary skill in the art would be motivated to utilize a separate file that can apply the same user edits to other model data without introducing errors during automatic generation of source code *(see Hunt – Paragraph [0004])*.

As per **Claim 12**, RTW_UG discloses:

- a processor for:
  - providing an automatic code generator to create source code that implements functionality of said at least one graphical model and that corresponds to data referenced by said at least one graphical model *(see Figure 3-1; Chapter 3-3, "The Real-Time Workshop automates the task of building a stand-alone program from your Simulink model.")*, and

- providing a predefined storage class specifying a first manner in which said automatic code generator creates said source code corresponding to said data referenced by said at least one graphical model in said modeling and execution environment *(see Chapter 3-6, "All pages of the Simulation Parameters dialog box affect Real-Time Workshop code generation."; Chapters 3-15 to 3-16, "There is an Options button on the Real-Time Workshop page of the Simulation Parameters dialog box. Pressing this button opens a Code Generation Options dialog box that varies depending on which system target file configuration you selected in the System Target File Browser."; Chapter 3-19, "Storage Class — you can change the storage class of the selected variable.")*; and

- a display device for:

- displaying a view of salient aspects of said source code generated by said automatic code generator *(see Section 11-6, "MdlStart Function Custom Code dialog box")*.

However, RTW_UG does not disclose:

- defining a custom storage class in said modeling and execution environment utilizing parameters selected by a user from a plurality of selectable parameters, said custom storage class specifying a second manner in which said automatic code generator creates source code corresponding to said data referenced by said at least one graphical model in said modeling and execution environment, said second manner differing from said first manner,

- generating source code implementing said functionality of said at least one graphical model using said automatic code generator, said generating using said custom storage class to generate source code corresponding to said data referenced by said at least one graphical model, and

- displaying a user interface with said plurality of selectable parameters for said custom storage class.

Hunt discloses:

- defining a custom storage class in a modeling and execution environment utilizing parameters selected by a user from a plurality of selectable parameters, said custom storage class specifying a second manner in which an automatic code generator creates source code corresponding to data referenced by at least one graphical model in said modeling and execution environment, said second manner differing from a first manner *(see Paragraph [0006], "An object is defined by a user entering meta data defining the object and the object's relationships with objects stored in the common object repository. Source code is created from the meta data, wherein the source code defines the object and the object's relationships."; Paragraph [0050], "Turning now to FIG. 1, by way of an overview, the user of the code generator 100 ... of the present invention enters meta-data 102 into the code generator 100's GUI."; Paragraph [0083], "Consider now the entry of the meta data used in connection with the present invention. Such information representing the meta data is input via the GUI such as illustrated in FIGS. 2-6."; Paragraph [0088], "When the user is ready to build the class, the "Generate" button 218 can be selected, for example by clicking with a mouse.")*,

- generating source code implementing said functionality of said at least one graphical model using said automatic code generator, said generating using said custom storage class to generate source code corresponding to said data referenced by said at least one graphical model *(see Paragraph [0092], "Referring now to FIG. 8, process 370 illustrates how the user interacts with the code builder tool 100 of the present invention starting at 380 where the code builder*

application is started. At 384 the builder dialogue 150 is instantiated to bring dialog 150 to the users display. The user selects an object [custom storage class] from the meta data at 388 and control passes to 392 where the user determines if the source code is to be created for the object. If the user chooses to create source code at this point (392), the source code is created at 396."), and

- displaying a user interface with said plurality of selectable parameters for said custom storage class *(see Figure 2; Paragraph [0049], "The code generator tool of the preferred embodiment of the present invention is embodied as an application program that presents the user with a Graphical User Interface (GUI) that can be used to easily input meta data about an object model.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to modify Real-Time Workshop® to include defining a custom storage class in said modeling and execution environment utilizing parameters selected by a user from a plurality of selectable parameters, said custom storage class specifying a second manner in which said automatic code generator creates source code corresponding to said data referenced by said at least one graphical model in said modeling and execution environment, said second manner differing from said first manner, generating source code implementing said functionality of said at least one graphical model using said automatic code generator, said generating using said custom storage class to generate source code corresponding to said data referenced by said at least one graphical model, and displaying a user interface with said plurality of selectable parameters for said custom storage class. The modification would be obvious because one of ordinary skill in the art would

be motivated to allow the user to define a custom storage class that applies the same user edits to model data without introducing errors during automatic generation of source code *(see Hunt – Paragraph [0004])*.

As per **Claim 13**, the rejection of **Claim 12** is incorporated; however, RTW_UG does not disclose:

- wherein said user-selected parameters for said custom storage class in said user interface are changed and said source code generated by said automatic code generator is adjusted to reflect said change in user-selected parameters.

Hunt discloses:

- wherein user-selected parameters for a custom storage class in a user interface are changed and source code generated by an automatic code generator is adjusted to reflect said change in user-selected parameters *(see Paragraph [0050], "When a change to the object model needs to be made, the user can simply update the meta data through the GUI and regenerate all the files associated with the object model.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hunt into the teaching of RTW_UG to modify Real-Time Workshop® to include wherein said user-selected parameters for said custom storage class in said user interface are changed and said source code generated by said automatic code generator is adjusted to reflect said change in user-selected parameters. The modification would be obvious because one of ordinary skill in the art would be motivated to ascertain that the

generated source code is concurrent with the most recent changes made by the user to the custom storage class parameters.

As per **Claim 14**, the rejection of **Claim 13** is incorporated; and RTW_UG further discloses:

- wherein said adjusted source code is displayed in said view of salient aspects of said source code *(see Chapter 11-6, "MdlStart Function Custom Code dialog box").*

As per **Claim 15**, the rejection of **Claim 12** is incorporated; and RTW_UG further discloses:

- wherein said view of salient aspects of said source code includes at least one token, said at least one token being symbolically representative of a non-displayed segment of code *(see Chapter 3-23, "Template makefiles are essentially makefiles in which certain tokens are expanded to create a makefile for your model (model.mk). model.mk is created from the template makefile specified in the Real-Time Workshop page of the Simulation Parameters dialog box by copying each line from the template makefile and expanding the tokens of Table 3-4.").*

Claims 16-23 and 26 are computer-readable medium claims corresponding to the method claims above (Claims 1-8 and 11) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1-8 and 11.

7. **Claims 9, 10, 24, and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over **RTW_UG** in view of **Hunt** as applied to Claims 1 and 16 above, and further in view of **US 2003/0225774 (hereinafter "Davidov")**.

As per **Claim 9**, the rejection of **Claim 1** is incorporated; however, RTW_UG and Hunt do not disclose:

- wherein said user-selected parameters include a non-portable directive to a compiler.

Davidov discloses:

- wherein user-selected parameters include a non-portable directive to a compiler *(see Paragraph [0092], "Command line options or directives for the compiler ..."; Paragraph [0214], "The compiler compiles Java source code produced by the generator according to supplied directives.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Davidov into the teaching of RTW_UG to modify Real-Time Workshop® to include wherein said user-selected parameters include a non-portable directive to a compiler. The modification would be obvious because one of ordinary skill in the art would be motivated to conveniently and dynamically create software programs that can be executed on a computer system.

As per **Claim 10**, the rejection of **Claim 9** is incorporated; however, RTW_UG and Hunt do not disclose:

- wherein said non-portable directive to a compiler assigns data to at least one memory location in said electronic device.

Davidov discloses:

- wherein a non-portable directive to a compiler assigns data to at least one memory location in an electronic device *(see Paragraph [0160], "The data is loaded when the application is started, and is saved when the application is destroyed. This type of persistence uses the device records management system (RMS), for example, non-volatile memory.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Davidov into the teaching of RTW_UG to modify Real-Time Workshop® to include wherein said non-portable directive to a compiler assigns data to at least one memory location in said electronic device. The modification would be obvious because one of ordinary skill in the art would be motivated to store data that can be utilized at a later time.

Claim 24 is rejected for the same reason set forth in the rejection of Claim 9.

Claim 25 is rejected for the same reason set forth in the rejection of Claim 10.

Response to Arguments

8. Applicant's arguments filed on April 21, 2009 have been fully considered, but they are not persuasive.

In the Remarks, Applicant argues:

a) RTW_UG does not disclose or suggest "predefined storage class specifying a first manner in which said automatic code generator creates said source code corresponding to said data referenced by said at least one graphical model … custom storage class specifying a second manner in which said automatic code generator creates source code corresponding to said data referenced by said at least one graphical model in said graphical modeling and execution environment, said second manner differing from said first manner," as recited in claim 1.

Examiner's response:

a) Examiner disagrees. With respect to the Applicant's assertion that RTW_UG does not disclose or suggest "providing a predefined storage class in said graphical modeling and execution environment, said predefined storage class specifying a first manner in which said automatic code generator creates said source code corresponding to said data referenced by said at least one graphical model in said graphical modeling and execution environment," the Examiner respectfully submits that RTW_UG clearly discloses "providing a predefined storage class in said graphical modeling and execution environment, said predefined storage class specifying a first manner in which said automatic code generator creates said source code corresponding to said data referenced by said at least one graphical model in said graphical modeling and execution environment" *(see Chapter 3-6, "All pages of the Simulation Parameters dialog box affect Real-Time Workshop code generation."; Chapters 3-15 to 3-16, "There is an Options button on the Real-Time Workshop page of the Simulation Parameters dialog box. Pressing this button opens a Code Generation Options dialog box that varies depending on which system target file configuration you selected in the System Target File*

Browser."; Chapter 3-19, "Storage Class — you can change the storage class of the selected variable."). Note that a user can choose a predefined storage class from a list of predefined storage classes to customize the Real-Time Workshop® code generation.

Therefore, for at least the reason set forth above, the rejections made under 35 U.S.C. § 103(a) with respect to Claims 1, 12, and 16 are proper and therefore, maintained.

In the Remarks, Applicant argues:

b) Hunt also does not disclose or suggest a custom storage class specifying a second manner in which an automatic code generator creates source code, as required by claim 1. The Hunt system does not support custom storage classes for creating source code corresponding to the objects in the Hunt repository. Therefore, both RTW_UG and Hunt fail to disclose or suggest a custom storage class specifying a second manner in which an automatic code generator generates source code, as required by claim 1.

Examiner's response:

b) Examiner disagrees. With respect to the Applicant's assertion that Hunt does not disclose or suggest a custom storage class specifying a second manner in which an automatic code generator creates source code, the Examiner respectfully submits that Hunt clearly discloses a custom storage class specifying a second manner in which an automatic code generator creates source code *(see Paragraph [0006], "An object is defined by a user entering meta data defining the object and the object's relationships with objects stored in the common object repository. Source code is created from the meta data, wherein the source code defines the object and the*

object's relationships."; Paragraph [0050], "Turning now to FIG. 1, by way of an overview, the user of the code generator 100 ... of the present invention enters meta-data 102 into the code generator 100's GUI."; Paragraph [0083], "Consider now the entry of the meta data used in connection with the present invention. Such information representing the meta data is input via the GUI such as illustrated in FIGS. 2-6."; Paragraph [0088], "When the user is ready to build the class, the "Generate" button 218 can be selected, for example by clicking with a mouse."). Note that Figures 2-6 of Hunt clearly illustrate a GUI that allows a user to input metadata for a customized class (custom storage class). Source code is generated from the metadata of the customized class using a code generator.

Therefore, for at least the reason set forth above, the rejections made under 35 U.S.C. § 103(a) with respect to Claims 1, 12, and 16 are proper and therefore, maintained.

Conclusion

9. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event,

however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

10. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Q. C./

Examiner, Art Unit 2191

/Wei Y Zhen/

Supervisory Patent Examiner, Art Unit 2191